Journal of King Saud University – Engineering Sciences (2014) 26, 11–19

King Saud University
Journal of King Saud University – Engineering Sciences
www.ksu.edu.sa
www.sciencedirect.com

ORIGINAL ARTICLE

A novel discrete particle swarm optimization for p-median problem

Mehmet Sevkli [a,*], Ruslan Mamedsaidov [b], Fatih Camci [c]

[a] King Saud University, Department of Industrial Engineering, P.O. Box 800, Riyadh 11421, Saudi Arabia
[b] Fatih University, Faculty of Engineering, Department of Industrial Engineering, Buyukcekmece, Istanbul 34500, Turkey
[c] Integrated Vehicle Health Management Centre, School of Applied Sciences Cranfield University, Cranfield, MK43 0FQ, United Kingdom

Received 30 May 2011; accepted 29 September 2012
Available online 9 October 2012

KEYWORDS
Particle swarm optimization;
p-Median problem;
Combinatorial optimization

Abstract p-Median problem is a well-known discrete optimization problem aiming to locate p number of facilities that satisfies the demand of multiple places with minimum cost. Even though continuous particle swarm optimization (PSO) has been successfully applied to many areas in recent years, discrete PSO algorithm is in its infancy. In this paper, a new discrete particle swarm optimization algorithm (NDPSO) is proposed for the p-median problem. Although presented algorithm has all major characteristics of the classical particle swarm optimization (PSO), its search strategy is different. The algorithm aims to minimize the distance between demand points and facilities. The algorithm has been tested on benchmarking problem instances from OR library and its performance is compared with other algorithms in the literature such as neural model, reduced variable neighborhood search, and simulated annealing. The presented method is also compared with two other existing discrete PSO algorithms in the literature. The experiments have shown that the proposed algorithm highly outperforms all the methods compared with better computational time.

1. Introduction

p-Median problem is a well-known facility-location problem where the task is to allocate p facilities in a way that the distance between n demand points and facilities is minimized. It is shown by Kariv and Hakimi (1979) that the p-median problem is NP-hard. It is unlikely to obtain an optimal solution through polynomial time-bounded algorithms. Small size instances of p-median problem can be solved with reasonable computational time by exact algorithms such as branch-and-bound Järvinen et al. (1972). This method finds (n–p) nodes that no facility is assigned, p-medians being the not found nodes. However, as the problem size increases, the computation time of exact methods increases exponentially. In contrast to exact methods, heuristic algorithms generally have acceptable time and memory requirements, but do not guarantee optimal solution. In these methods, a feasible solution is obtained which is likely to be either optimal or near optimal.

* Corresponding author. Address: King Saud University, College of Engineering, Industrial Engineering Department, Office: 2A127/2, P.O. Box 800, Riyadh 11421, Saudi Arabia. Tel.: +966 1 4676826.
E-mail addresses: msevkli@ksu.edu.sa, msevkli@gmail.com (M. Sevkli).
Peer review under responsibility of King Saud University.

ELSEVIER Production and hosting by Elsevier

1018-3639 © 2012 Production and hosting by Elsevier B.V. on behalf of King Saud University.
http://dx.doi.org/10.1016/j.jksues.2012.09.002

One of the first heuristic methods applied to the p-median problem is a greedy heuristic by Kuehn and Hamburger (1963). Avella et al. (2012) introduced a new heuristic for large-scale PMP instances, based on Lagrangean relaxation. Another heuristic method applied to p-median problem was proposed by (Teitz and Bart (1968)).

In addition to heuristic methods, metaheuristic algorithms are also studied for p-median problem. The most popular Metaheuristics that appeared in the literature was presented by Alp et al. (2003). In this method, a Genetic Algorithm, where the gene of a chromosome consists of indexes of nodes that were selected as facility locations, is presented. (Murray and Church (1996)) applied simulated annealing to location-planning models. More recently Levanova and Loresh (2004) implemented the ant systems and simulated annealing algorithm to solve p-median problem. Alcaraz et al. (2012) proposed different configurations of two hybrid metaheuristics to solve the problem, a genetic algorithm and a scatter search approach. Cadenas et al. (2011) used genetic algorithm for the fuzzy p-median problem. Crainic et al. (2003, 2004) used variable neighborhood search. More information about p-median problem can be found in Mladenovic et al. (2007) and experimental evaluation of the complexity of the p-median problem can be found at (Goldengorin and Krushinsky, 2011). In this paper, we propose a discrete particle swarm optimization (PSO) algorithm for p-median problem. The novelty of this study comes from the success of the algorithm as it is the first implementation of PSO for p-median to our knowledge.

PSO is based on the metaphor of social interaction and communication among different spaces in nature, such as bird flocking and fish schooling. It is different from other evolutionary methods in a way that it does not use the genetic operators (such as crossover and mutation), and the members of the entire population are maintained throughout the search procedure. Thus, information is socially shared among individuals to direct the search toward the best position in the search space. In a PSO algorithm, each member is called a particle, and each particle moves around in the multi-dimensional search space with a velocity constantly updated by the particle's experience, the experience of the particle's neighbors, and the experience of the whole swarm. PSO was first introduced to optimize various continuous nonlinear functions by Eberhart and Kennedy, (1995). PSO has been successfully applied to a wide range of applications such as automated drilling (Onwubolu and Clerc, 2004), neural network training (Van den Bergh and Engelbecht, 2000), scheduling problems ((Tasgetiren et al., 2006); Tseng and Liao, 2008; Allahverdi and Al-Anzi, 2006) and uncapacitated facility location problems (Sevkli and Guner, 2008).

In addition, PSO algorithm was successfully applied to the binary problems, but researchers have not reached to a general consensus on details of Binary PSO (BPSO). First binary PSO was introduced by Kennedy and Eberhart (1997), in which velocity is defined as the chance of changing from 1 to 0 or vice versa. Authors mentioned about the unsatisfactory answers of some questions in JKennedy et al. (2001). In recent years, various new velocity and position change formulations are presented Afshinmanesh et al., 2005; Khanesar et al., 2007; Camci, 2009. More information about PSO can be found in Kennedy et al. (2001).

In this paper, a novel discrete particle swarm optimization algorithm (NDPSO) is proposed for the p-median problems. Although the algorithm has all major characteristics of the classical PSO, the search strategy of the algorithm is different. The NDPSO is applied to the p-median problems with the objective of minimizing distance between demand points and facilities. This paper reports the comparison results of presented method with the algorithm proposed by Domínguez and Muñoz (2008), simple simulated annealing (SA), reduced variable neighborhood search (RVNS) algorithm, and the BPSO methods (Afshinmanesh et al., 2005; (Khanesar et al., 2007) presented in the papers above.

The organization of this paper is as follows: Section II introduces p-median problem, mathematical representation of the problem is shown and an overview of classical PSO and the various BPSO algorithms. The third section reveals the proposed heuristic algorithm. The implementations of simple SA and RVNS are elaborated in Section 3 while experimental results and comparison with other methods are reported and discussed in Section 4. Finally, the fifth section includes the concluding remarks.

2. Background

2.1. Problem description

In the p-median problem p facilities are to be allocated among n demand points in such a way that the distance from the demand point to the nearest facility is minimized.

The mathematical formulation for p-median problem is as follows (ReVelle and Swain, 1970):

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} a_i d_{ij} x_{ij} \tag{1}$$

Subject to:

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, 2, \ldots, n, \tag{2}$$

$$x_{ij} \leqslant y_j \quad i,j = 1, 2, \ldots n, \tag{3}$$

$$\sum_{j=1}^{n} y_j = p \tag{4}$$

$$x_{ij}, y_j \in \{0, 1\} \quad i,j = 1, 2, \ldots, n, \tag{5}$$

where n = total number of nodes in the graph, a_i = demand of node i, d_{ij} = distance from node i to node j, p = number of facilities used as medians, and a_i, dij are positive real numbers,

$$x_{ij} = \begin{cases} 1 & \text{if node } i \text{ is assigned to facility at point } j, \\ 0 & \text{Otherwise} \end{cases} \tag{6}$$

$$Y_j = \begin{cases} 1 & \text{if facility is locate at point } j, \\ 0 & \text{Otherwise} \end{cases} \tag{7}$$

2.2. Classic particle swarm optimization

In PSO, each single solution, called a particle, is considered as an individual; the group becomes a swarm (population) and the search space is the area to explore. Each particle has a

fitness value calculated by a fitness function and a velocity to fly toward the optimum. All particles fly across the problem space following the particle that is nearest to the optimum. PSO starts with an initial population of solutions, which is updated iteration-by-iteration. The principles that govern PSO algorithm can be stated as follows:

- n dimensional position $(X_i = (x_{i1}, x_{i2}, ..., x_{in}))$ and velocity vector $(V_i = (v_{i1}, v_{i2}, ..., v_{in})$ for i^{th} particle starts with a random position and velocity.
- Each particle knows its position and value of the objective function for that position. The best position of i^{th} particle is denated as $P_i = (p_{i1}, p_{i2}, ..., p_{in})$, and the best position of the whole swarm as, $G = (g_1, g_2, ..., g_n)$ respectively. The PSO algorithm is governed by the following main equations:

$$v_{in}^{t+1} = wv_{in}^{t} + c_1 r_1 (p_{in}^{t} - x_{in}^{t}) + c_2 r_2 (g_i^{t} - x_{in}^{t}), x_{in}^{t+1} = v_{in}^{t+1} + x_{in}^{t} \quad (8)$$

where t represents the iteration number, w is the inertia weight which is a coefficient to control the impact of the previous velocities on the current velocity. c_1 and c_2 are called learning factors. r_1 and r_2 are uniformly distributed random variables in [0, 1] .

The original PSO algorithm can optimize problems, in which the elements of the solution space are continuous real numbers. The major obstacle for successfully applying PSO to combinatorial problems in the literature is its continuous nature. To remedy this drawback, Tasgetiren et al. (2006) presented the smallest position value (SPV) rule. Another approach to tackle combinatorial problems with PSO is done by Pan et al. (2008). They generate a similar PSO equation to update the particle's velocity and position vectors using one and two cut genetic crossover operators. A discrete particle swarm optimization was proposed by utilizing a probability value in a range of [0.0, 0.1] according to which the position vector takes discrete values, 0/1, (Kennedy and Eberhart, 1997)). Different versions of velocity and position update formulations have been proposed in Afshinmanesh et al. (2005) and Khanesar et al. (2007). In Afshinmanesh et al. (2005), velocity and position updates are calculated using logical operators such as "and", "or", "xor", etc., as in following equations:

$$\begin{aligned} v_{i,t+1} &= or(and(rand(1,m), xor(g_i^t, x_i^t)), \\ &\quad \text{and}(\text{rand}(1,m), \text{xor}(G, x_i^t)))x_i^{t+1} \\ &= \text{xor}(x_t^i, v_i^t) \end{aligned} \quad (9)$$

This method will be referred as BPSO1 in the rest of the paper. Different velocity and location calculations are presented in Khanesar et al. (2007) as shown in the formulation below. This method will be referred as BPSO2 in the rest of the paper. The comparison of these methods on maintenance optimization problem is presented in Camci (2009).

$$x_{i,j}^{t+1} = \begin{cases} \overline{x_{i,j}^t} \; if \; r_{i,j} < V_{i,j} \\ x_{i,j}^t \text{otherwise} \end{cases} V_{i,j} = \begin{cases} V_{i,j}^0 if x_{i,j} = 1 \\ V_{i,j}^1 if x_{i,j} = 0 \end{cases} \quad (10)$$

3. A novel discrete particle swarm optimization algorithm

In this study, a novel discrete particle swarm optimization algorithm (NDPSO) is proposed for the p-median problem. This algorithm starts with a randomly generated initial population and the position and fitness value of each individual is assigned to its personal best. The best individual in the whole swarm with its position and fitness value, on the other hand, is assigned as the global best. Then, the position of each particle is updated based on the personal best and the global best for every iteration. The main concepts of NDPSO are similar to classical PSO algorithm; however, the search strategy is different. That is, each particle in the swarm moves based on the following equations.

$$\begin{aligned} s_1 = w^t \oplus \eta(X_i^t), w^{t+1} = w \cdot \beta, s_2 = c_1 \oplus \eta(P_i^t), s_3 \\ = c_2 \oplus \eta(G^t), X_i^{t+1} = \text{best}(s_1; s_2; s_3) \end{aligned} \quad (11)$$

At each iteration, the position vector of each particle, its personal best and the global best are considered. Randomly selected particles are updated by applying *exchange* function (η). A random number between 0 and 1 is generated for each particle, and the ones with smaller random number than user-defined inertia weights (w) are selected.

Exchange function (η) implies the exchange of randomly chosen facility with another randomly chosen node. In other words, a node that has facility is randomly selected and the facility in this node is carried to a randomly selected other node that do not have the facility. For instance, for the p-median problem of $p = 4$, suppose a sequence of {5, 11, 17, 32} is a possible solution set. In order to apply *exchange* function, we also need to derive two random numbers; one is for determining the facility to be changed and the other is for the node to be accepted as a new facility. Let us say those numbers are 11 and 20 (that is, the node number 11 will not serve as a facility anymore and the node number 20 will be accepted as a new facility {5, *11*, 17, 32}). The new sequence will be {5, *20*, 17, 32}. Note that the order of the facilities shown in the solution set is not important. In other words the solution set can be shown in another way too {32, 20, 17, 5}.

After the particle is manipulated with this *exchange* function (say the resulting solution is s_1) the inertia weight is discounted by a constant factor (β) at each iteration. This reduces the number of randomly selected particles for manipulation as the algorithm in late iterations compared to early iterations.

The next step is to generate another random number of U (0, 1) to be compared with c_1, cognitive parameter, to make a decision whether to apply *exchange* function to personal best of the particle considered. If the random number is less than c_1, then the personal best of the particle undertaken is manipulated and the resulting solution is spared as s_2. Likewise, a third random number of U (0, 1) is generated for making a decision whether to manipulate the global best with the *exchange* function. If the random number is less than c_2, social parameter, then *exchange* is applied to the global best to obtain a new solution of s_3. Unlike the case of inertia weight, the values of c_1 and c_2 factors are not increased or decreased iteratively, but are fixed at 0.5. That means the probability of applying *exchange* function to the personal and global bests remains the same. The new replacement solution is selected among s_1, s_2 and s_3, based on their fitness values. This solution may not always be better than the current solution. This is to keep the swarm diverse. The convergence is traced by checking the personal best of each new particle and the global best. As it is seen, proposed equations have all major characteristics of

```
Begin
 Initialize particles (population) randomly
 For each particle
   Calculate fitness value
   Set to position vector and fitness value as personal best (P_i^t)
   Select the best particle and its position vector as global best(G_t)
 End
 Do{
   Update inertia weight
   For each particle
    Apply Exchange with the probability of inertia weight (s_1)
    Apply Exchange to (P_i^t) with the probability of c_1 (s_2)
    Apply Exchange to (G_t) with the probability of c_2 (s_3)
    Select the best one among the s_1,s_2 and s_3
    Update personal best (P_i^t)
   End
   Update global best (G_t)
 }While (Maximum Iteration is not reached)
End
```

Figure 1 Pseudo code of the NDPSO algorithm for p-median problem.

the classical PSO equations. The following pseudo-code describes in detail the steps of the NDPSO algorithm. (See Figure 1).

4. Implementations reduced variable neighborhood search and simulated annealing algorithms

In this section, we introduce the implementation of reduced variable neighborhood search (RVNS) and simple simulated annealing (SA) algorithms for the p-median problem in order to use in comparison study.

4.1. Reduced variable neighborhood search

Variable neighborhood search (VNS) is one of the most recent metaheuristics and known as one of the very well-known local search methods (Mladenovic and Hansen, 1997), that gets more attention day-by-day, because of its ease of use and accomplishments in solving combinatorial optimization problems. Basically, a local search algorithm carries out exploration within a limited region of the whole search space. That facilitates a provision of finding better solutions without going into further investigation. The VNS is a simple and effective search procedure that proceeds to a systematic change of neighborhood. An ordinary VNS algorithm gets an initial solution, xS, where S is the whole set of search space, than manipulates it through a two nested loop in which the core one alters and explores via two main functions the so called shake and local search. The outer loop works as a refresher reiterating the inner loop, while the inner loop carries the major search. Local search explores an improved solution within the local neighborhood, while shake diversifies the solution by switching to another local neighborhood. The inner loop iterates as long as it keeps improving the solutions, where an integer, k, controls the length of the loop. Once an inner loop is completed, the outer loop re-iterates until the termination condition is met. Since the complementariness of neighborhood functions is the key idea behind VNS, the neighborhood

```
Procedure RVNS
    Generate initial Solution, s∈S
    kmax←2
    while termination condition not met do
     k←1
     while(k≤kmax) do
      if(k=1) then s′∈S←Exchange(s)
      if(k=2) then s′∈S←Swap(s)
      if (f(s′)≤f(s)) then s← s′
      else k←k+1
      end-if
     end-while
     if((f(s′)≤f(s)) then s← s′
    end-while
end-procedure
```

Figure 2 Pseudo code of RVNS algorithm for p-median problem.

structure/heuristic functions should be chosen very rigorously so as to achieve an efficient VNS.

Reduced variable neighborhood search (RVNS) is a variation of the VNS. In VNS, the solution space is searched with a systematic change of neighborhood. It has two main steps named *LocalSearch* and *Shake*. VNS becomes RVNS if *LocalSearch* step is removed. RVNS is usually preferred as a general optimization method for problems where exhaustive local search is expensive.

The implemented pseudo-code of RVNS is given in Figure 2. Both the choice and the order of neighborhood structures are critical for the performance of the RVNS algorithm. Exchange and Swap moves are taken as neighborhood structure. Starting from a solution and the first neighborhood (exchange(s)), during each iteration of the inner loop, a random solution s' is selected from the current neighborhood. If s' is better than s, it replaces s and the search continues with the first neighborhood. Otherwise, the algorithm switches to the

Table 1 NDPSO and NA-L + performances for OR library test problems.

Problem	(n, p)	Optimum	Objective function		% error		CPU time (s)	
			NDPSO	NA–L +	NDPSO	NA–L +	NDPSO	NA–L +
pmed1	(100, 5)	5819	5819	5819	0	0	0, 56	0, 27
pmed2	(100, 10)	4093	4099	4093	0, 15	0	21, 60	1, 86
pmed3	(100, 10)	4250	4250	4250	0	0	2, 02	0, 83
pmed4	(100, 20)	3034	3034	3038	0	0, 13	11, 84	21, 37
pmed5	(100, 33)	1355	1356, 5	1359	0, 11	0, 3	33, 79	27, 46
pmed6	(200, 5)	7824	7824	*****	0	*****	1, 37	*****
pmed7	(200, 10)	5631	5631	5631	0	0	18,72	7,37
pmed8	(200, 20)	4445	4445	4448	0	0,07	21, 06	77, 5
pmed9	(200, 40)	2734	2740, 5	2751	0, 24	0, 62	119, 44	120, 89
pmed10	(200, 67)	1255	1256, 5	1264	0, 12	0, 72	105, 80	167, 07
pmed11	(300, 5)	7696	7696	7696	0	0	2, 04	3, 23
pmed12	(300, 10)	6634	6634	*****	0	*****	5,13	*****
pmed13	(300, 30)	4374	4374	*****	0	*****	42, 93	*****
pmed14	(300, 60)	2968	2972, 9	2983	0, 17	0, 51	144, 05	388, 51
pmed15	(300, 100)	1729	1733, 7	1751	0, 27	1, 27	150, 24	526, 11
pmed16	(400, 5)	8162	8162	*****	0	*****	6, 36	*****
pmed17	(400, 10)	6999	7000, 2	6999	0, 02	0	57, 98	94, 02
pmed18	(400, 40)	4809	4817, 5	4811	0, 18	0,04	150,48	787, 03
pmed19	(400, 80)	2845	2863, 7	2863	0, 66	0, 63	158,84	1024, 5
pmed20	(400, 133)	1789	1805,4	1815	0, 92	1, 45	293, 66	1317
pmed21	(500, 5)	9138	9138	*****	0	*****	4, 24	*****
pmed22	(500, 10)	8579	8579	*****	0	*****	33, 00	*****
pmed23	(500, 50)	4619	4650, 6	4624	0, 68	0, 11	155, 87	1889, 5
pmed24	(500, 100)	2961	2989, 6	2986	0, 97	0,84	394, 75	2157, 2
pmed25	(500, 167)	1828	1845, 3	1865	0, 95	2, 02	727, 81	2634,6
pmed26	(600, 5)	9917	9917	*****	0	*****	20, 79	*****
pmed27	(600, 10)	8307	8307, 9	8307	0, 01	0	74, 44	171, 75
pmed28	(600, 60)	4498	4539, 6	4508	0, 92	0, 22	260, 62	3368, 7
pmed29	(600, 120)	3033	3062	3060	0, 96	0,89	708, 85	3827, 8
pmed30	(600, 200)	1989	2007, 5	2016	0, 93	1, 36	1972, 30	4705, 3
pmed31	(700, 5)	10086	10086, 1	*****	0	*****	53, 87	*****
pmed32	(700, 10)	9297	9297, 4	*****	0	*****	77, 96	*****
pmed33	(700, 70)	4700	4744	4706	0, 94	0, 13	451, 27	5927, 2
pmed34	(700, 140)	3013	3042, 2	3038	0, 97	0, 83	1303, 58	6747, 6
pmed35	(800, 5)	10400	10400	*****	0	*****	25, 38	*****
pmed36	(800, 10)	9934	9945, 5	*****	0, 12	*****	113, 79	*****
pmed37	(800, 80)	5057	5104, 7	5071	0, 94	0, 28	948, 08	9243, 1
pmed38	(900, 5)	11060	11060	*****	0	*****	29, 19	*****
pmed39	(900, 10)	9423	9423	*****	0	*****	111, 10	*****
pmed40	(900, 90)	5128	5177, 8	5155	0, 97	0, 53	1393, 01	13331

*****, The results were not presented in published work.

next neighborhood structure (swap(s)). If all neighborhood structures are exhausted, the inner loop is initiated all over again starting from the first neighborhood. The outer loop is repeated until a stopping condition is met.

4.2. Simulated annealing

Simulated annealing (SA) was developed in 1983 by Kirkpatrick et al. (1983) to deal with highly nonlinear problems. It works by emulating the physical process whereby a solid slowly cooled so that when eventually its structure is frozen, this happens at a minimum energy configuration. By analogy with this physical process, each step of the SA algorithm attempts to replace the current solution by a random solution. The new solution may then be accepted with a probability that depends both on the difference between the corresponding function values and temperature (T), that is gradually decreased during the process. All the generation and acceptance depend on the T.

A brief implementation of simple SA algorithm for the p-median problems is as follows. A random initial solution is created. The new solution is generated by an *exchange* function aforementioned above. The new solution is accepted if its solution is better than the original solution; otherwise, it is accepted with some probability which decreases as the process evolves. The most common function form of the acceptance probability is $exp(-\Delta/T)$, where Δ is the increase in the objective function and T is the control parameter (temperature). Initially temperature is set to a high level so that a neighborhood exchange happens frequently in early iterations. It is gradually lowered using a predetermined cooling schedule in later iterations to make exchanges only when a better solution is obtained. The final solution is found until a stopping condition is met.

Table 2 NDPSO and NA–L + performances for OR library test problems.

Problem	(n, p)	Optimum	Average percentile deviation			Minimum percentile deviation			Maximum percentile deviation		
			RVNS	SA	NDPSO	RVNS	SA	NDPSO	RVNS	SA	NDPSO
pmed1	(100, 5)	5819	0, 18	0, 26	0	0	0	0	0, 60	1, 12	0
pmed2	(100, 10)	4093	1,11	1, 38	0, 15	0, 17	0, 29	0	2, 71	2, 96	0, 29
pmed3	(100, 10)	4250	1, 71	1, 20	0	0	0	0	3, 67	3, 67	0
pmed4	(100, 20)	3034	3, 30	3, 90	0	1, 45	0, 96	0	6, 43	6, 36	0
pmed5	(100, 33)	1355	4,49	4, 63	0, 11	2, 51	3,03	0	8, 63	8,56	0,22
pmed6	(200, 5)	7824	0,20	0, 90	0	0	0, 12	0	0, 55	1,94	0
pmed7	(200, 10)	5631	1, 75	1, 75	0	0, 66	0, 64	0	3, 46	4, 49	0
pmed8	(200, 20)	4445	4, 56	4, 53	0	2, 72	3,24	0	5, 96	6,57	0
pmed9	(200, 40)	2734	5, 68	6, 27	0, 24	3, 44	3, 69	0	7, 50	7, 72	0, 62
pmed10	(200, 67)	1255	7, 34	9, 03	0, 12	3, 35	5, 82	0	10, 12	15, 14	0, 40
pmed11	(300, 5)	7696	0, 53	1, 12	0	0	0, 35	0	1,53	2, 29	0
pmed12	(300, 10)	6634	2, 27	2, 64	0	1, 39	1, 45	0	4, 16	6, 21	0
pmed13	(300, 30)	4374	5, 09	5, 97	0	2, 95	3, 06	0	6, 93	7,84	0
pmed14	(300, 60)	2968	7, 23	8, 02	0, 17	6, 10	6, 77	0	8, 66	8, 96	0, 47
pmed15	(300, 100)	1729	8,95	10, 67	0, 27	6, 54	7, 98	0,06	11, 57	14, 23	0, 52
pmed16	(400, 5)	8162	0,84	1, 30	0	0, 01	0, 29	0	2, 28	2,96	0
pmed17	(400, 10)	6999	3, 29	3, 32	0, 02	1, 70	1, 19	0	5, 33	4, 91	0, 06
pmed18	(400, 40)	4809	5, 32	6,33	0, 18	3, 78	4, 53	0,04	6, 49	7, 49	0, 48
pmed19	(400, 80)	2845	7,82	9, 65	0, 66	6,15	7, 21	0,35	9, 00	11, 63	0, 98
pmed20	(400, 133)	1789	11,03	14, 38	0, 92	9, 39	10, 96	0,84	13, 47	22, 30	0, 95
pmed21	(500, 5)	9138	1, 72	3, 01	0	0	1, 26	0	3, 30	5, 37	0
pmed22	(500, 10)	8579	2, 56	3, 36	0	1,08	2, 30	0	4, 57	5, 47	0
pmed23	(500, 50)	4619	7, 00	7, 50	0, 68	5, 59	5, 72	0,26	7, 64	9, 37	0, 97
pmed24	(500, 100)	2961	8,88	10, 58	0, 97	7, 16	8, 48	0,91	10, 37	13, 41	0,98
pmed25	(500, 167)	1828	11,87	15,79	0, 95	9, 52	13, 02	0,82	13, 02	19, 58	0, 98
pmed26	(600, 5)	9917	1, 66	2, 56	0	0, 30	1, 01	0	2, 86	4, 13	0
pmed27	(600, 10)	8307	3, 13	3, 47	0, 01	1, 66	2, 19	0	4, 78	5, 51	0,04
pmed28	(600, 60)	4498	7,14	7, 55	0, 92	6, 07	5, 96	0,78	8, 14	9, 52	0,98
pmed29	(600, 120)	3033	10,53	12, 03	0, 96	8,90	9, 89	0,86	11,61	16, 16	0, 99
pmed30	(600, 200)	1989	12, 80	16, 51	0, 93	11, 31	13, 88	0,80	14, 88	21, 72	0, 96
pmed31	(700, 5)	10086	1, 53	1, 98	0	0, 35	0, 20	0	3, 54	3, 39	0, 01
pmed32	(700, 10)	9297	3, 27	3, 72	0	2, 07	2, 59	0	5, 16	4, 88	0,04
pmed33	(700, 70)	4700	7, 34	9, 70	0, 94	5, 47	8, 11	0,83	8, 57	13, 06	0, 98
pmed34	(700, 140)	3013	11, 78	14, 23	0, 97	10, 36	12, 58	0,93	13, 84	18, 39	1,00
pmed35	(800, 5)	10400	1, 64	2, 47	0	0, 01	0, 53	0	3, 90	4, 06	0
pmed36	(800, 10)	9934	2, 82	3, 11	0, 12	1, 80	2, 18	0	4, 02	4, 59	0, 31
pmed37	(800, 80)	5057	8, 40	9, 40	0, 94	7, 10	7, 53	0,83	9, 55	11, 94	0, 99
pmed38	(900, 5)	11060	1,65	1, 79	0	0, 80	0, 37	0	2, 88	2, 73	0
pmed39	(900, 10)	9423	3, 02	4,07	0	1, 89	3, 41	0	4, 75	6, 02	0
pmed40	(900, 90)	5128	8, 84	10, 22	0, 97	7, 61	8, 39	0, 90	10, 04	11, 91	0, 99
Average values			5, 01	6, 01	0, 45	3, 53	4, 28	0, 23	6, 66	8, 46	0, 38

Table 3 Comparison of BPSO1, BPSO2, and NDPSO with minimum value obtained from ten runs.

(n, p)	Optimum	Objective function			% error		
		NDPSO	BPSO1	BPSO2	NDPSO	BPSO1	BPSO2
Minimum							
(100, 5)	5819	5819	5821	5819	0	0, 03	0, 00
(100, 10)	4093	4099	4113	4093	0, 15	0, 49	0, 15
(100, 10)	4250	4250	4351	4257	0	2, 38	0, 00
(100, 20)	3034	3034	3163	3034	0	4, 25	0, 00
(100, 33)	1355	1356,5	1402	1355	0, 11	3, 47	0, 11
(200, 5)	7824	7824	7962	7914	0	1, 76	0, 00
(200, 10)	5631	5631	5742	5675	0	1, 97	0, 00
(200, 20)	4445	4445	4705	4514	0	5, 85	0, 00
(200, 40)	2734	2740,5	2897	2792	0, 24	5, 96	0, 24
(200, 67)	1255	1256,5	1383	1293	0, 12	10, 20	0, 12
Average					0, 06	3, 64	0, 06

Table 4 Comparison of BPSO1, BPSO2, and NDPSO with maximum value obtained from ten runs.

(n, p)	Optimum	Objective function			% error		
		NDPSO	BPSO1	BPSO2	NDPSO	BPSO1	BPSO2
Maximum							
(100, 5)	5819	5819	6006	5819	0	3, 21	0, 00
(100, 10)	4093	4099	4417	4117	0, 15	7, 92	0, 59
(100, 10)	4250	4250	4676	4324	0	10, 02	1, 74
(100, 20)	3034	3034	3279	3053	0	8, 08	0, 63
(100, 33)	1355	1356, 5	1517	1372	0, 11	11, 96	1, 25
(200, 5)	7824	7824	8405	8043	0	7, 43	2, 80
(200, 10)	5631	5631	6001	5755	0	6, 57	2, 20
(200, 20)	4445	4445	4922	4633	0	10, 73	4, 23
(200, 40)	2734	2740, 5	3060	2882	0, 24	11, 92	5, 41
(200, 67)	1255	1256, 5	1466	1368	0, 12	16, 81	9, 00
Average					0, 06	9, 46	2, 79

5. Experimental results

In this section, a comparison study is carried out on the effectiveness of the proposed NDPSO. NDPSO was exclusively tested in comparison with the result of Enrique Domínguez and José Muñoz (Mladenovic and Hansen, 1997), simple simulated annealing (SA) and reduced variable neighborhood search (RVNS) algorithms. A data set of 40 p-median problems with known optimal solutions in the OR library (Beasley, 1990) was used in the testing of performances of algorithms. NDPSO, SA and RVNS algorithms were coded in C and run on a PC with the configuration of 2.6 GHz CPU and 512 MB memory. The size of the population considered by all algorithms is twice the number of nodes. The algorithms were run for 1000 generations. The initial temperature and the decrement factor β (cooling schedule) are taken as 100 and 0.095 respectively for the simple SA algorithm.

Relative percentile deviation is taken as a performance measure to compare the performance of the algorithms.

$$\frac{f - f_{\text{opt}}}{f_{\text{opt}}} \cdot 100$$

where f denotes the best solution found by the algorithm and f_{opt} denotes the optimum value of the objective function.

For NDPSO, the social and cognitive parameters were taken as $c_1 = c_2 = 0.5$, initial inertia weight is set to 0.5, and the decrement factor β is fixed at 0.9995.

Table 1 summarizes the comparison of computational results for 10 replications of NDPSO and neural algorithm by Enrique Domínguez and José Muñoz (NA–L+) Pan et al., 2008 solving 40 test problems from OR library. The objective function values found by the algorithms are shown in columns 4 and 5. The % error is shown in columns 6 and 7 and the computational time is shown in columns 8 and 9 respectively.

Table 2 summarizes the comparison of computational results for 10 replications of NDPSO to coded simple simulated annealing and reduced variable neighborhood search algorithm, solving 40 test problems from OR library. The average percentile deviation at each algorithm is shown in columns 4, 5 and 6. The minimum percentile deviation at each algorithm is shown in columns 7, 8 and 9 and maximum percentile deviation at each algorithm is shown in columns 10, 11 and 12 respectively.

From the tables given you can see that the proposed NDPSO algorithm performs better not only in terms of the percentile error but it finds the "near optimal" solution in less computational time for some of the problems.

Table 5 Comparison of BPSO1, BPSO2, and NDPSO with average value obtained from ten runs.

(n, p)	Optimum	Objective function			% error		
		NDPSO	BPSO1	BPSO2	NDPSO	BPSO1	BPSO2
Average							
(100, 5)	5819	5819	5907, 3	5819	0	1, 52	0, 00
(100, 10)	4093	4099	4257, 5	4099, 8	0,15	4, 02	0, 17
(100, 10)	4250	4250	4456, 3	4291, 2	0	4, 85	0, 97
(100, 20)	3034	3034	3233, 8	3038, 3	0	6, 59	0, 14
(100, 33)	1355	1356, 5	1464, 4	1360, 5	0,11	8, 07	0, 41
(200, 5)	7824	7824	8220, 7	7992, 5	0	5, 07	2, 15
(200, 10)	5631	5631	5873, 4	5712, 1	0	4, 30	1, 44
(200, 20)	4445	4445	4793, 1	4558, 5	0	7, 83	2, 55
(200, 40)	2734	2740, 5	3005	2841, 2	0,24	9, 91	3, 92
(200, 67)	1255	1256, 5	1429, 5	1322, 3	0,12	13, 90	5, 36
Average					0, 06	6, 61	1,71

Table 6 Comparison of computational times of BPSO1, BPSO2, and NDPSO.

CPU time (s)		
NDPSO	BPSO1	BPSO2
0, 56	7, 80	77, 97
21, 60	9, 48	78, 40
2, 02	9, 25	78, 95
11, 84	11, 30	78, 96
33, 79	11, 45	76, 21
1, 37	26, 98	314, 90
18, 72	40, 29	350, 47
21, 06	47, 82	330, 27
119, 44	64, 30	341, 34
105, 80	77, 94	343, 81
Average		
33, 62	30, 66	207, 13

Table 3 displays the minimum value obtained from ten runs for presented method and BPSO algorithms in the literature. Tables 4 and 5 display the maximum and average values of the same ten runs. Tables also include the relative percentile deviation given above. As seen from the tables, the presented method highly outperforms the methods in the literature. Table 6 shows the computational times of these methods. The computational time of the presented method is also better.

6. Conclusion

The main objective in this work is to allocate p facilities in a way that the total distance between n demand points and facilities is minimized. This is a well-known facility-location problem in the literature called as a p-median problem. It is unlikely to obtain an optimal sequence of points through polynomial time-bounded algorithms. On the other hand, heuristic algorithms generally have acceptable time and memory requirements, but do not guarantee optimal solution. They give a feasible solution, which is likely to be either optimal or near optimal. Particle swarm optimization (PSO) is one of the latest metaheuristic methods in the literature. Even though PSO initially assumes continuous variables, recent studies report different versions of discrete PSO algorithms. In this paper a novel proposed discrete particle swarm optimization (NDPSO) algorithm is proposed. The algorithm has been tested on benchmarking datasets from OR library and compared with other algorithms in literature. The results show that the proposed algorithm highly outperforms the other algorithms.

In summary, the results presented in this work are very encouraging and promising for the application of the NDPSO to p-median problem. For the future work, this NPSO algorithm can be applied to the other combinatorial optimization problems such as scheduling and location problems.

References

Afshinmanesh, F., Marandi, A., Rahimi-Kian, A., 2005. A Novel Binary Particle Swarm Optimization Method Using Artificial Immune System. EUROCON 2005, Serbia & Montenegro, Belgrade.

Alcaraz, J., Landete, M., Monge, J.F., 2012. Design and analysis of hybrid metaheuristics for the reliability p-median problem. European Journal of Operational Research 222 (1), 54–64.

Allahverdi, A., Al-Anzi, F.S., 2006. Evolutionary heuristics and an algorithm for the two-stage assembly scheduling problem to minimize makespan with setup times. International Journal of Production Research 44 (22), 4713–4735.

Alp, O., Erkut, E., Drezner, Z., 2003. An efficient genetic algorithm for the p-median problem. Annals of Operations Research 122, 21–42.

Avella, P., Boccia, M., Salerno, S., Vasilyev, I., 2012. An aggregation heuristic for large scale p-median problem. Computers & Operations Research 39 (7), 1625–1632.

Beasley, J.E., 1990. OR-Library: distributing test problems by electronic mail. Journal of the Operational Research Society 41 (11), 1069–1072.

Cadenas, J.M., Canós, M.J., Garrido, M.C., Ivorra, C., Liern, C., 2011. Soft-computing based heuristics for location on networks: the p-median problem. Applied Soft Computing 11 (2), 1540–1547.

Camci, F., 2009. Comparison of genetic and binary particle swarm optimization algorithms on system maintenance scheduling using prognostics information. Engineering Optimization 41 (2), 119–136.

Crainic, T.G., Gendreau, M., Hansen, P., Mladenovic, N., (2003). Parallel variable neighborhood search for the p-median, Les Cahiers du GERAD, G-2003-4.

Crainic, T.G., Gendreau, M., Hansen, P., Mladenovic, N., 2004. Cooperative parallel variable neighborhood search for the p-median. Journal of Heuristics 10 (3), 293–314.

Domínguez, Enrique., Muñoz, José, 2008. A neural model for the p-median problem. Computers & Operations Research 35, 404–416.

Eberhart, R.C., Kennedy, J., (1995). A new optimizer using particle swarm theory, Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 39–43.

Goldengorin, B., Krushinsky, D., 2011. Complexity evaluation of benchmark instances for the p-median problem. Mathematical and Computer Modelling 53 (9–10), 1719–1736.

Järvinen, P., Rajala, J., Sinervo, H., 1972. A branch-and-bound algorithm for seeking the p-median. Operations Research 20 (1), 173–178.

Kariv, O., Hakimi, S.L., 1979. An algorithmic approach to network location problems. II. The p-medians. SIAM Journal on Applied Mathematics 37 (3), 539–560.

Kennedy, J., Eberhart, R.C. (1997). A discrete binary version of the particle swarm algorithm. Proceedings of the International Conference on Systems, Man, Cybernetics. Piscataway, NJ, 4104–4109.

Kennedy, J., Eberhart, R.C., Shi, Y., 2001. Swarm Intelligence. San Mateo. Morgan Kaufmann CA, USA.

Khanesar, M.A., Teshnehlab, M., Shoorehdeli, M.A., (2007). A Novel Binary Particle Swarm Optimization, Mediterranean Conference on Control and Automation, July 27–29 2007, Athens, Greece.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by Simulated Annealing. Science. New Series 220 (4598), 671–680.

Kuehn, A.A., Hamburger, M.J., 1963. A heuristic program for locating warehouses. Management Science 9 (4), 643–666.

Levanova, T.V., Loresh, M.A., 2004. Algorithms of ant system and simulated annealing for the p-median problem. Automation and Remote Control 65 (3), 431–438.

Mladenovic, N., Hansen, P., 1997. Variable neighborhood search. Computers & Operations Research 24 (11), 1097–1100.

Mladenovic, N., Brimberg, J., Hansen, P., Moreno-Pérez, J.A., 2007. The p-median problem: a survey of metaheuristic approaches. European Journal of Operational Research 179, 927–939.

Murray, A.T., Church, R.L., 1996. Applying simulated annealing to location-planning models. Journal of Heuristics 2, 31–53.

Onwubolu, G.C., Clerc, M., 2004. Optimal operating path for automated drilling operations by a new heuristic approach using particle swarm optimization. International Journal of Production Research 42 (3), 473–491.

Pan, Q.-K., Tasgetiren, M.F., Liang, Y.-C., 2008. A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. Computers & Operations Research 35 (9), 2807–2839.

ReVelle, C., Swain, R., 1970. Central facilities location. Geographical Analysis 2, 30–42.

Sevkli, M., Guner, A.R., 2008. A discrete particle swarm optimization algorithm for uncapacitated facility location problem. Journal of Artificial Evolution and Applications 861512, 1–9.

Tasgetiren, M.F., Liang, Y.-C., Sevkli, M., Gencyilmaz, G., 2006. Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem. International Journal of Production Research 44 (22), 4737–4754.

Teitz, M.B., Bart, P., 1968. Heuristic methods for estimating the generalized vertex median of a weighted graph. Operations Research 16 (5), 955–961.

Tseng, C.T., Liao, C.J., 2008. A particle swarm optimization algorithm for hybrid flow-shop scheduling with multiprocessor tasks. International Journal of Production Research 46 (17), 4655–4670.

Van den Bergh, F., Engelbecht, A.P., 2000. Cooperative learning in neural networks using particle swarm optimizers. South African Computer Journal 26, 84–90.